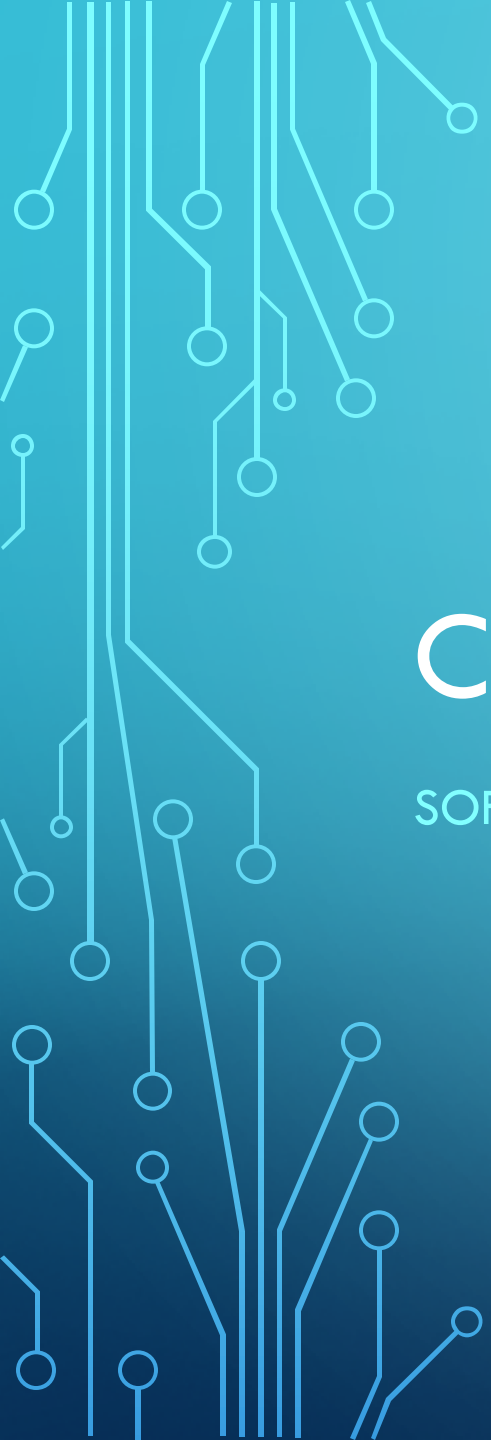


CS370

SOFTWARE METHODOLOGY

A BIT ABOUT ME…

- SSU Graduate, Class of 1993
- 24 Years at Autodesk
  - Build and Integrations
  - ARX API group
  - ObjectDBX
  - Vertical applications for GIS, Process & Power, Mechanical Engineering
  - Software Engineering Manager
- Currently at Salesforce, in Technical Writing
  - I love to explain things.

THESE ARE THE PEOPLE IN YOUR NEIGHBORHOOD

TYPICAL ROLES IN THE SOFTWARE DEVELOPMENT CYCLE

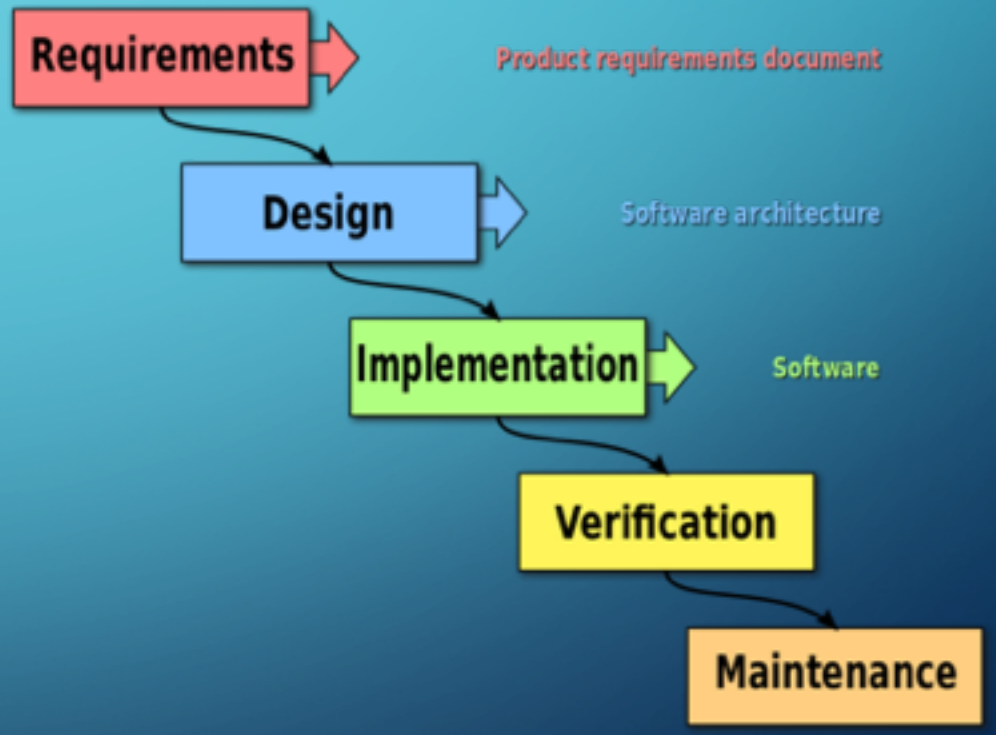
- Product Manager
  - Business role. Identifies customer needs. Defines requirements.
- Product Designer
  - Defines look and feel, UX/UI, workflows.
- Architect
  - An experienced software engineer. Helps the software team clarify what form the code should take, especially when adding to an existing system.
- Software Engineer
  - Writes, tests, and documents code.

- Testers
  - Everyone tests. But testers can have specialized skills, or be responsible for testing frameworks. Testers typically exercise a broader workflow than SWD.
- Documentation
  - Help docs, API docs, validates UI text, and more!
- Localization
  - Translates the product and the documentation to different languages.
- Operations
  - Responsible for source code control systems and build automation.
- Project Manager
  - Keeps all of the above on track. Reports on team progress.

METHODOLOGY – WATERFALL (OLD WAY)

Originally conceptualized in the 1970s, Waterfall is a sequential methodology used in project management, but applied specifically over the years to Software Development. The name is derived from the conceptual process, which is envisioned as a downward flow through the different phases.

No later phase may be started until its immediate predecessor has been completed to the satisfaction of the stakeholders.



REQUIREMENTS SPECIFICATION PHASE

The definition of all requirements specifications which culminates in the product requirements document (PRD). The PRD has a specific format that typically contains functional, usability, technical, support, and interaction requirements.

The Product Manager typically produces this document as a result of customer research.

- Functional Requirements: What the product should do (feature functionality).
- Usability Requirements: How the product should allow users to access feature functionality.
- Technical Requirements: Addressing concerns regarding platform, security, minimum system requirements, integration with other systems, and distribution.
- Support Requirements: Typically an SLA (Service Level Agreement) stating the level of support that a product will receive before and after launch. Before launch, this may be user training. After launch, this is represent by service packs addressing previously undiscovered bugs.
- Interaction Requirements: Representing the integration of the software with other existing systems.

UX/UI DESIGN PHASE (ALSO REQUIREMENTS)

The definition of all UX/UI requirements, which typically culminates in the creation of a Design Specification. The Design Spec has detailed information about dialogs, workflows, and interactions.

The Designer typically produces this document based on the PRD, in consultation with the Product Manager, and on user research.

SOFTWARE DESIGN PHASE

- The phase wherein the architecture for the software is formulated based on the requirements specified in the PRD and Design Specification. This phase is also used to define minimum hardware and system requirements for running the software.

- The output of this phase is typically a High Level Software Design (HLSD) document, describing software components, data, and interfaces.

- Data – what information needs to be stored, and how accessible must it be, to achieve the workflow?
- Architecture – what classes must be created or extended to achieve the workflow? How shall they be organized; new libraries, extending existing code, factoring existing code and extracting?
- What existing features and code might be impacted by these changes? What risks exist?
- What functionality needs to be exposed to third party developers as APIs?
- How can the code be made performant (quick)?
- How long will the new code take to implement?

IMPLEMENTATION PHASE

- Wherein the coding of the actual software takes place. The system is first developed in functionally distinct areas called units. Each unit is developed and tested for its functionality. (Unit testing.)

INTEGRATION AND TESTING PHASE

- Wherein all functional units development in the implementation phase are individually tested and integrated into a system (the software solution as a whole). Post integration, the entire system is tested for defects as a unit. Bugs are discovered and resolved.
- Documentation and Localization also occur during this stage.

DELIVERY PHASE

- Once the functional and non functional testing is done, the product is deployed in the customer environment or released to the market.

- Deployment is its own engineering challenge.

MAINTENANCE PHASE

- Defects may be discovered in the client environment which require patch releases. To enhance the product, improved versions are released. Maintenance is done to deliver these changes to the customer environment.

WATERFALL RELEASE CYCLE

- Waterfall stages are typically executed within a fixed schedule, with a known target release date.
- Commonly:
  - 20% - 40% for the initial stages.
  - 30%-40% for coding.
  - Remainder for testing, integration, documentation and delivery.

WHY WATERFALL?

- Easy to understand and manage.
- Emphasis on documentation ensures knowledge of the overall deliverable is well understood, and easy to transfer to new team members.
- Careful planning can save time and resources later.
- Works well when project requirements are unlikely to change.
- “Do it once, do it right.”

GAME 1 - BATTLESHIP

Place 3 ships on your Fleet area: 1 cruiser, 2 battleships

Group A:

Standard rules: Take turns announcing a target to your opponent, “D4”. Announce “hit” if your opponent guessed a square occupied by your ship, otherwise announce “miss.”

Mark hits with an X and misses with a slash on the Targeting area. If all the squares of your ship have been hit, announce, “You sunk my battleship.”

Group B:

Modified rules: Place your ships, then number spaces in your targeting area from 1 to 50.

Follow the standard rules, but you must call out the targets in the order that you numbered them.

WHY NOT WATERFALL?

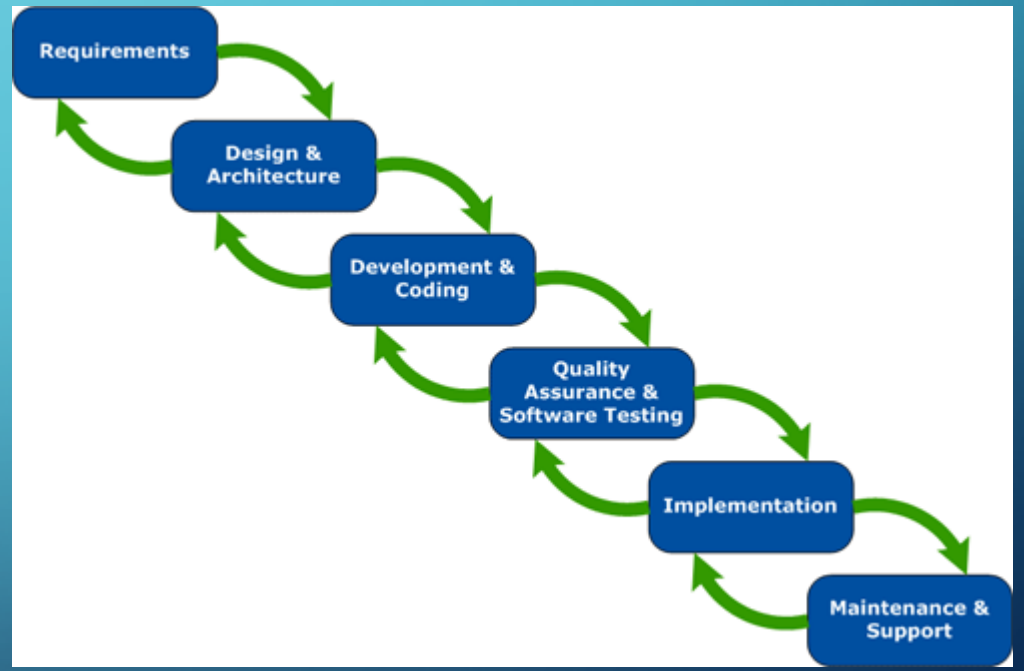
- Doesn't allow you to react to change.
- Hard to complete the early phases, especially if customers can't articulate their needs.
- Unforeseen challenges in implementation are common, but schedules are fixed.
- Functional software doesn't exist until late in the cycle, making meaningful feedback come too late to be acted upon.
- Leaving system testing to the end means that meaningful bugs aren't found until fixing them has become risky.

FROM THE TRENCHES…

- Daily work varies from non-existent to overwhelming.
- Requirements and Design occur before engineering input, which can result in a feature set that cannot be completed in the remaining time.
- Scope creep: User input and discoveries continue to occur during the coding cycle, and feature changes are requested. This impacts development schedule.
- The handoff pattern leads to adversarial relationships between the roles.

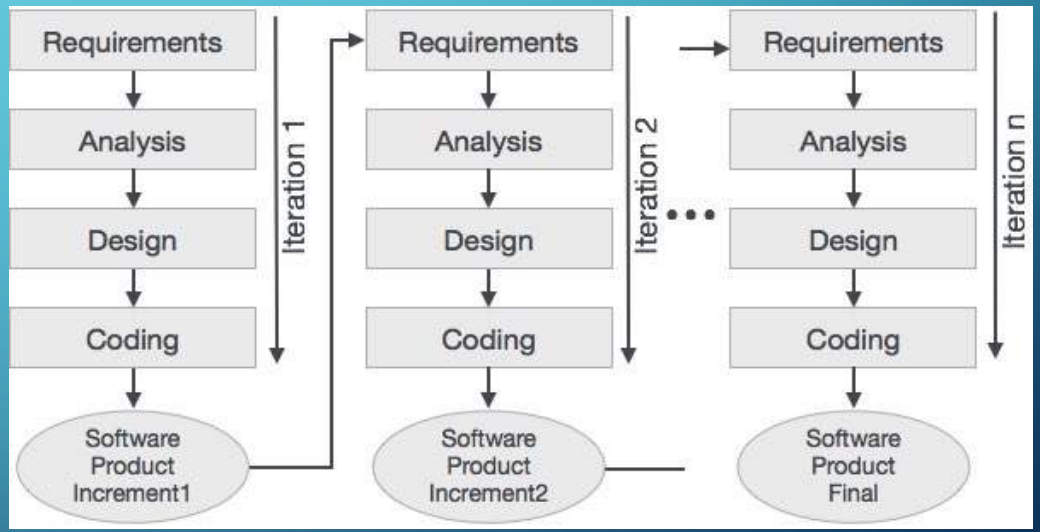
HOW TO MAKE WATERFALL BETTER?

- Step 1: Add more overlap and feedback between the stages. "Sashimi" model.



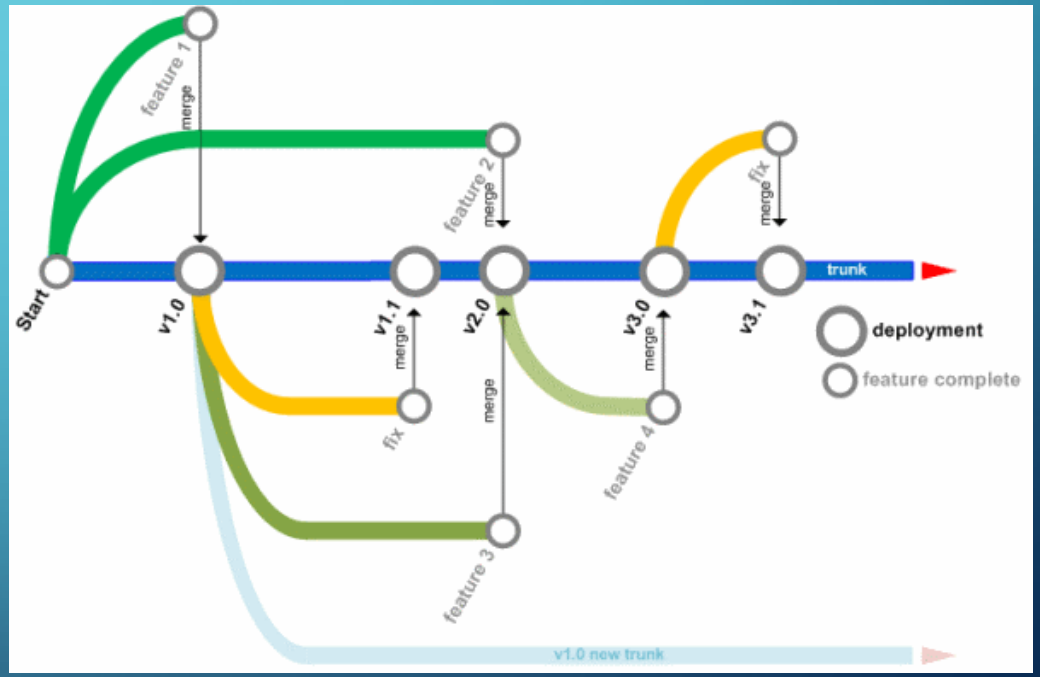
HOW TO MAKE WATERFALL BETTER?

- Step 2: Break down development into smaller, iterative cycles that complete a subset of the requirements and are fully tested.



HOW TO MAKE WATERFALL BETTER?

- Step 3: Utilize source code control branches to separate feature work, and redirect incomplete features to the next release.



AGILE… THE NEXT BIG THING

- Incorporates LEAN principles from the automotive industry, such as:
  - Eliminate the cost of late changes by not making irreversible decisions too early.
  - Wait until you know what the customer wants, and then deliver it as quickly as possible.

- Incremental improvements to the waterfall model were also leading in this direction…

Agile software development is a conceptual group of methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible responses to change.

THE AGILE MANIFESTO

We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value:

- **Individuals and interactions** over Processes and Tools
- **Working Software** over Comprehensive Documentation
- **Customer Collaboration** over Contract Negotiation
- **Responding to Change** over Following a Plan

That is, while there is value in the items on the right, we value the items on the left more.

- Customer satisfaction by rapid delivery of useful software.
- Welcome changing requirements, even late in development.
- Working software is delivered frequently (weeks, not months or years).
- Close, daily cooperation between business people and developers
- Projects are built around motivated individuals, who should be trusted
- Face-to-face conversation is the best form of communication (co-location)
- Working software is the principle measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount work not done—is essential
- Self-organizing teams
- Regular adaptation to changing circumstances.

SCRUM

Conceptualized in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka, scrum is an iterative and incremental agile software development framework for managing product development. It works outside of the traditional, cyclic approach and recognizes that changing requirements are a regular factor in software development.

Scrum uses an ‘empirical approach,’ which accepts that a given problem (requirement) can never be fully understood or defined, and instead focuses on allowing a development team to deliver rapid iterations as new details of a requirement emerge.

SCRUM OVERVIEW

- Roles: The Product Owner, the Scrum Master, and the Scrum Team
- Rituals: The Daily Standup, Sprint Planning, Sprint Review, Retrospective
- Artifacts: The Product Backlog, the Sprint Backlog, the Burndown, and the product itself.

SCRUM ROLES

Product Owner: The product owner represents the stakeholders and is the voice of the customer. He or she is accountable for ensuring that the team delivers value to the business.

The Product Owner is responsible for the composition of customer desired items (typically user stories), ranks and prioritizes them, and adds them to the product backlog. Scrum teams should have one Product Owner, and while they may also be a member of the development team, this role should not be combined with that of the Scrum Master.

Communication is a main function of the product owner. Product owners bridge the communication gap between the team and their stakeholders.

- Demonstrates the solution to key stakeholders who were not present in a normal iteration demo.
- Announces releases.
- Communicates team status.
- Organizes milestone reviews.
- Educates stakeholders in the development process.
- Negotiates priorities, scope, funding, and schedule.
- Ensures that the Product Backlog is visible, transparent, and clear.
- Signs off on completed user stories.

SCRUM ROLES

Scrum Master: Scrum is facilitated by a Scrum Master, who is accountable for removing anything blocking the ability of the team to achieve their project goals and deliverables. The Scrum Master is not a traditional team lead or project manager, acting instead as a buffering layer to protect team members from distraction (churn).

The Scrum Master ensures that the Scrum process is used as intended. The Scrum Master is the enforcer of the rules of Scrum, often chairs key meetings, and challenges the team to improve.

Duties include:

- Helping the Product Owner maintain the product backlog in a way that ensures the project is well defined and the team can continually advance forward on the project at any given time
- Determine the definition of done for the project with input from the entire Scrum team
- Coaching the team within the Scrum principles in order to reach the defined 'done'
- Promote self-organization within the team
- Remove all impediments to the team's progress, both internal and external to the Scrum team
- Facilitate team meetings to ensure regular progress
- Educate others involved with the project on Scrum principles

SCRUM ROLES

Scrum Team: The Scrum Team is responsible for delivering potentially shippable increments implemented software at the end of each sprint (the Sprint Goal).

A Team is made up of 3–9 individuals responsible for the various aspects of implementation (analysis, design, development, test, technical communication, documentation, etc.). The Scrum Team is self-organizing, even though there may be some level of interface with project management.

DEFINING WORK – THE USER STORY

The **User Story** takes the form: As a <role>, I want <solution>, so that I can <value>.

"As a user, I want to upload photos to a website, so I can share them with friends."

"As an administrator, I want to approve photos, so that I can ensure they are appropriate."

Before work starts on a user story, the team will discuss it several times, decide how much work is required, and break large stories down into smaller stories.

PRIORITIZING WORK – THE PRODUCT BACKLOG

Product Backlog: Consisting of an ordered list of requirements that is recorded and maintained for a software development product. A Product Backlog is comprised of features, bug fixes, and non-functional requirements (NFR). It is intended to act as a high level record of the work required to complete and successfully deliver an MVP.

The Product Backlog Items (PBI) are organized according to priority by the Product Owner. The considerations made by a PO should be made around:

- Risk: Does the effort or impediments to a particular backlog item require it to be lower in priority until the risk can be mitigated?
- Business Value: Does the feature represent a higher or lower value in terms of meeting user and business requirements?
- Schedule: Is there a driving date that requires a particular item to be higher in priority?
- While the backlog and the relative value of the PBIs is the responsibility of the PO, the development team is responsible for sizing (determining the level of effort required to implement) the item.

The process of ordering the Product Backlog is called **Grooming**.

PREPARING TO WORK

- Story **Elaboration** – In which the team meets to clarify the details of the user story, to ensure everyone has the same understanding, and resolve all unanswered questions, so that development can proceed.
- **Definition of Ready**
  - Defined clearly enough that all members of the team understand what must be done.
  - Clear acceptance criteria.
  - Clear statement of business value.
  - Includes any required enabling specs, wireframes, etc.
  - Meets INVEST criteria.
  - Free from external dependencies (work beyond the team's control)
  - PITFALL: Too strict a DoR can prevent work from starting.

PREPARING TO WORK -- INVEST

- Independent – can be freely reordered in the backlog
- Negotiable/Negotiated – The concrete solution is negotiated by the team, who can enhance or rewrite the story.
- Valuable/Vertical – Adds something useful for the end user. A working slice of front end, scripts & DB, not a finished DB without a front end.
- Estimable – If you can't estimate, you need to break the story into pieces, get a better understanding, or explore the unknowns.
- Small – Smaller is easier to estimate, easier to understand, and easier to test. Fewer misunderstandings hide in small stories.
- Testable – You have to be able to test the story to know if it's done. If you can't think of a test, the story is still too fuzzy, and needs more elaboration.

PREPARING TO WORK – THE MVP

MVP – Minimum Viable Product

- A development technique in which a product or website is developed with sufficient features to satisfy early adopters. The final complete set of features is designed and developed after getting feedback from initial users.
- Stresses the impact of learning in new product development.

PREPARING TO WORK - ESTIMATION

Exercise 2 – How big is an elephant?

	Weight in Pounds		Relative size – light to heavy	
Canada Goose				
Rainbow Trout				
Asian Elephant				
Hummingbird				
Shetland Pony				
Polar Bear				
German Shephard				
Blue Whale				

PREPARING TO WORK – THE STORY POINT

- The **story point** is an **abstract** measure of effort to implement a story
- Takes into account:
  - The amount of work
  - The complexity of work
  - Risks or unknowns
  - The Definition of Done
- Usually follows a modified Fibonacci sequence: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100
- Anchor story – as the team gets experienced in estimating this way, an anchor story may emerge that grounds future estimations.

PREPARING TO WORK – WHEN YOU JUST DON'T KNOW

The **SPIKE** – A story or task aimed at gathering information, rather than producing a shippable product.

- A spike should have a clear objective.
- A spike should be timeboxed.
- The results must be reported to the team.

A spike can be used to do proof-of-concept work, or other experimentation.

Trouble estimating, or high point estimates, may indicate a spike is needed.

PREPARING TO WORK – SPRINT PLANNING

In which the team collectively story points the well-elaborated stories at the top of the product backlog, and commits to completing certain stories in a fixed time.

The stories that the team agrees to implement become the **Sprint Backlog**.

DOING THE WORK

- Exercise 3 – The Name Game
- Round 1: The Shout
- Round 2: Everyone's Happy?
- Round 3: Focus

DOING THE WORK – THE SPRINT

- AKA **Iteration**, a sprint is an uninterrupted* period of development.
- Typically 1 to 4 weeks. (2 is common.)
- Once started, no changes are allowed that would endanger the sprint goal.
- Quality goals are fixed.
- Some negotiation in scope is allowed as a result of learning.
- The outcome of the sprint is a **potentially shippable** product.

* Regular planning meetings and sprint rituals occur during this time.

DOING THE WORK – THE DAILY STANDUP

In which team members update each other on progress.

This is a short, daily meeting, typically 15 minutes or less. It is held standing, to keep everyone focused and on topic. Team members report:

- Yesterday's progress
- Today's plan
- Anything blocking their work

The need to collaborate on a problem or gather additional understanding from the PO often surfaces in the daily standup, and is addressed in a separate, ad-hoc, meeting.

Co-location enables quick discussion and resolution of daily discoveries.

Show up on time.

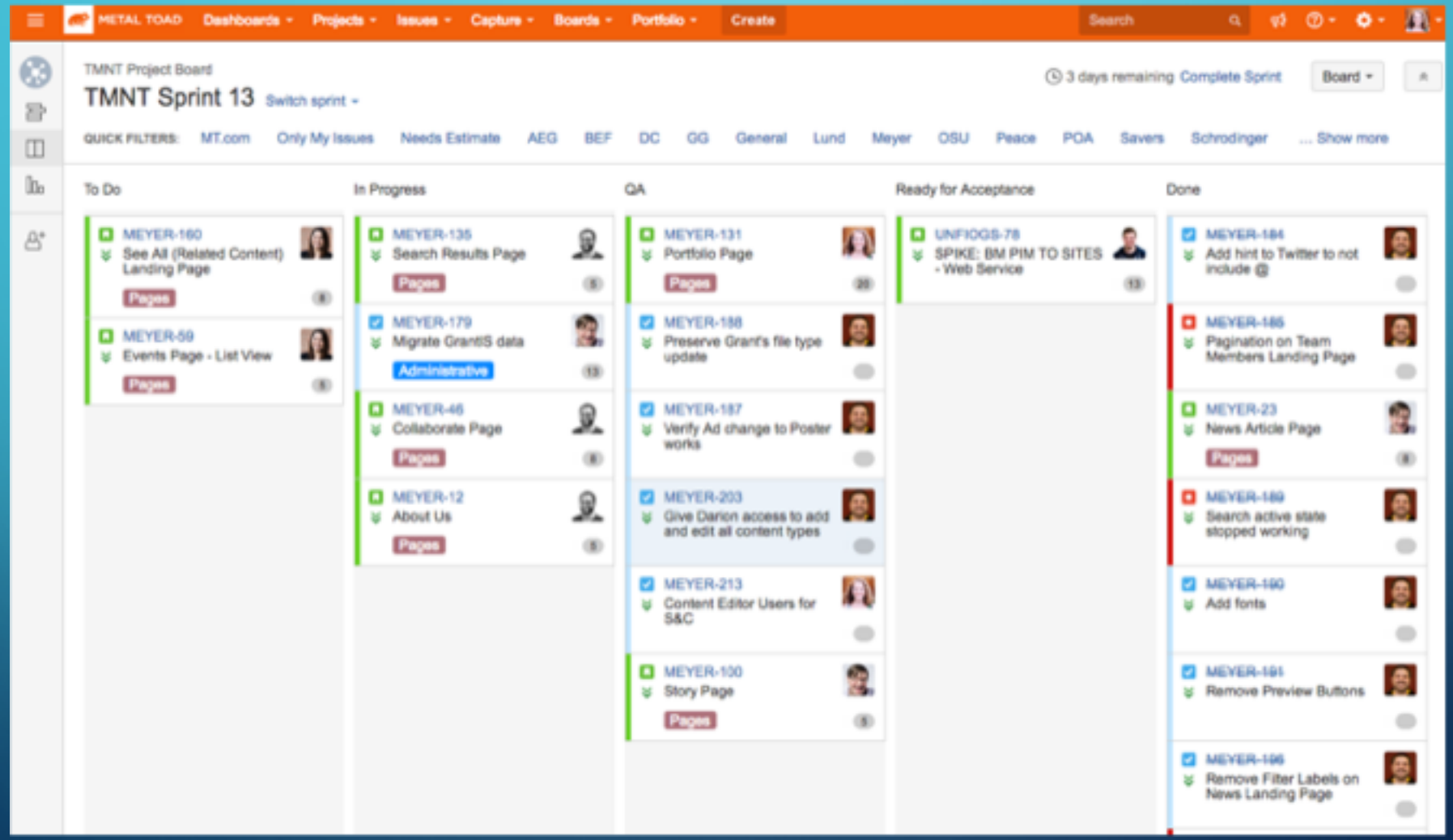
DOING THE WORK – THE DEFINITION OF DONE

Just because it compiles, doesn't mean you're done.

- Is it releasable?
- Does it meet the acceptance criteria?
- Is it unit/integration tested?
- Are the automations written?
- Are the release notes written?
- Is it properly globalized?
- No new technical debt?

The team may adjust the DoD.

SCRUM BOARD


METAL TOAD
Dashboards
Projects
Issues
Capture
Boards
Portfolio
Create
Search
TMNT Project Board
TMNT Sprint 13
Switch sprint
3 days remaining
Complete Sprint
Board
QUICK FILTERS:
MT.com
Only My Issues
Needs Estimate
AEG
BEF
DC
GG
General
Lund
Meyer
OSU
Peace
POA
Savers
Schrodinger
... Show more
To Do
In Progress
QA
Ready for Acceptance
Done
MEYER-160
See All (Related Content) Landing Page
Pages
MEYER-59
Events Page - List View
Pages
MEYER-135
Search Results Page
Pages
MEYER-179
Migrate GrantS data
Administrative
MEYER-46
Collaborate Page
Pages
MEYER-12
About Us
Pages
MEYER-131
Portfolio Page
Pages
MEYER-188
Preserve Grant's file type update
MEYER-187
Verify Ad change to Poster works
MEYER-203
Give Darion access to add and edit all content types
MEYER-213
Content Editor Users for S&C
MEYER-100
Story Page
Pages
UNFIOGS-78
SPIKE: BM PIM TO SITES - Web Service
MEYER-184
Add hint to Twitter to not include @
MEYER-185
Pagination on Team Members Landing Page
MEYER-23
News Article Page
Pages
MEYER-189
Search active state stopped working
MEYER-190
Add fonts
MEYER-191
Remove Preview Buttons
MEYER-196
Remove Filter Labels on News Landing Page

IMPROVING THE WORK – THE SPRINT REVIEW

In which the results of the sprint are demonstrated.

Sprint reviews are generally kept informal, and prefer live demos of new software over slide presentations.

The Product Owner should always attend. The project is assessed against the overall goal of the sprint.

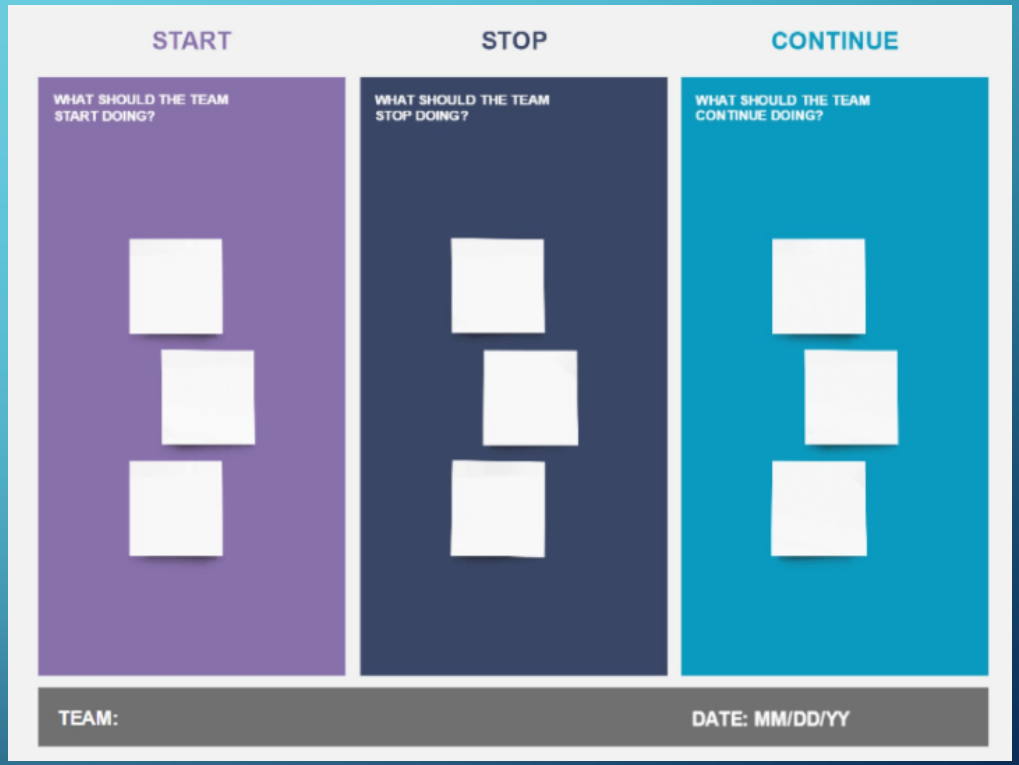
In a large enterprise, many teams demo to each other.

Usually, the review is FUN. It's exciting to see new work as it emerges.

IMPROVING THE WORK – THE RETROSPECTIVE

In which the team assesses its performance and looks for opportunities to improve.

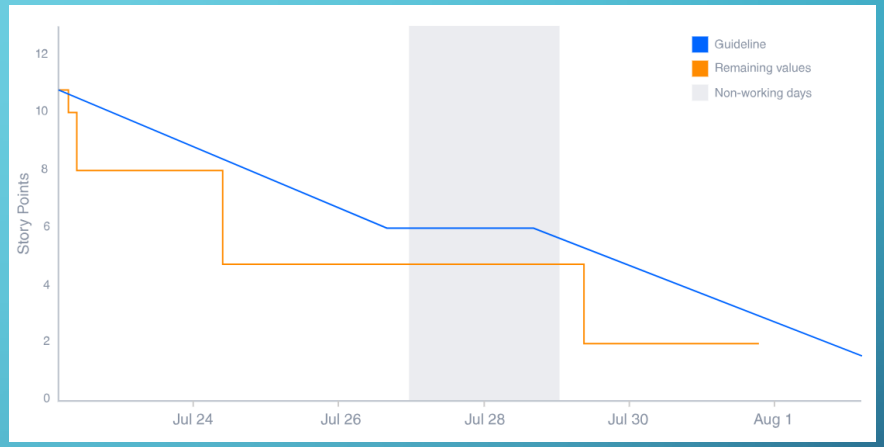
- Often held right after the sprint review.
- Golden rule: Assume everyone is doing their best and coming from a place of good intent. Then explore how to make things better.
- Typical Format: Things that worked (keep doing), Things that didn't work (stop doing), things to change (start doing).
- Commit to trying at least one change from the retrospective in the next sprint, then revisit.

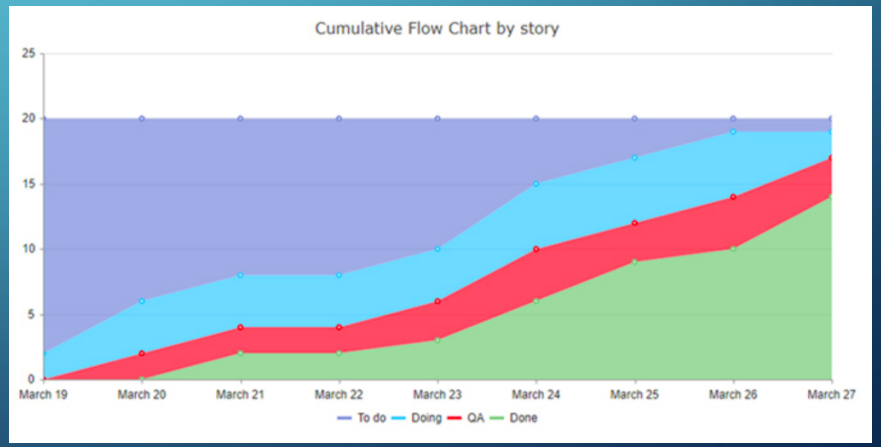


IMPROVING THE WORK – METRICS

- Burndown Charts
- Cumulative Flow Diagram
- Velocity

These tools are for the TEAM.





IMPROVING THE WORK

Exercise 4 – The Ball Point Game

Rules:

- Everyone must touch the ball
- Ball must have air time
- No ball to your direct neighbor
- Start point = end point
- Iteration = 2 minutes
- Retrospective = 1 minute
- 5 iterations

MAKING AGILE DEMAND DRIVEN - KANBAN

- Visualize what you do today (workflow): see all items in context
- Limit the amount of WIP: balance the flow and don't overcommit.
- Enhance flow: when something is done, take the next highest thing on the backlog to work on.

KANBAN VS. SCRUM

- No prescribed roles
- Continuous delivery
- Work is pulled through the system piece by piece
- Changes can be made at any time
- Cycle time
- Works when priorities are rapidly shifting

- PO, SM, and Team roles
- Timeboxed sprints
- Work goes through the system in batches (sprint backlog)
- No changes mid-sprint
- Velocity
- Appropriate where work can be prioritized in batches

PARTING THOUGHTS – MAKING AGILE BIG

- The Scaled Agile Framework (SAFe) is a methodology to help enterprises scale up agile methodologies.
  - Scrum of Scrums
  - Alignment, transparency are key principles

PARTING THOUGHTS – MAKING AGILE WORK

- Agile requires top-down understanding and commitment.
  - Commit to non-deterministic shipping dates.
  - Let the teams perform.
- Agile requires investment in supporting infrastructure and teams:
  - Robust OPS group that can support CI/CD with branched development.
  - Robust test automation system that is sufficiently parallelized to run after every build.
  - Robust delivery system/team that can deploy any build.